MISB MOTION IMAGERY STANDARDS BOARD	MISB RP 0811
Recommended Practice	**14 May 2009**
JPIP Profile (Client/Server Functions)	

1 Scope

This Recommended Practice (RP) defines a JPEG 2000 Interactive Protocol (JPIP) Profile for client/server interaction. It defines the expected behavior for client/server interactions for the delivery of JPEG 2000 compressed imagery within the context of the JPIP protocol. It does not address the delivery of metadata elements within JPIP. While JPIP does provide mechanisms for the delivery of metadata within a JPIP client/server exchange, it is envisioned that other alternate methods (e.g. OGC's WFS protocol, XML documents, etc.) will be preferred for the delivery of metadata.

It is expected that the JPIP protocol will find use within Large Volume Streaming Data (LVSD) systems (also known as WALF, WAPS, WAAS, etc.). JPIP is useful for any system that collects large frame imagery and compresses it using the JPEG 2000 standard and wishes to disseminate that imagery over reduced bandwidth links. In LVSD systems, JPIP could be used to serve imagery off of the collecting platform while a mission is in progress. JPIP might also be used to serve large imagery out of ground stations and libraries.

The Motion Imagery Standards Board recommends JPIP as a delivery protocol for LVSD imagery for platforms and libraries that collect and store this data. JPIP is applicable only to JPEG 2000 compressed imagery (it is not appropriate for MPEG compressed video) and not all LVSD systems may support real-time or near real-time delivery during collects. Therefore, JPIP is not mandated for LVSD systems. If systems do elect to use JPIP, this Recommended Practice provides guidance on the proper implementation of the protocol.

2 References

Normative References:

The following documents contain provisions that, through reference in this text, constitute provisions of the RP 0811. Applicability is limited to only the specific instance of the reference document; other aspects of referenced documents are for information. At the time of publication the editions indicated were valid but all documents are subject to revision. Parties in agreement, based on this profile, are warned against automatically applying more recent editions of the documents listed in this section. The nature of references made by the profile to such documents is specific to a particular edition. Members of IEC and ISO maintain a register of currently valid International Standards and profiles.

ISO/IEC 15444-1, *Information Technology — JPEG 2000 image coding system: Core coding system*, 2004.

ISO/IEC 15444-1:2004/Amd 1, *Profiles for digital cinema applications*, 2006.

ISO/IEC 15444-1:2004/Cor 1, *Information technology — JPEG 2000 image coding system: Core coding system TECHNICAL CORRIGENDUM 1*, 2007.

ISO/IEC 15444-2, *JPEG 2000 Image Coding System — Part 2: Extensions*, 2004.

ISO/IEC 15444-2:2004/Cor 3 *JPEG 2000 Image Coding System — Part 2: Extensions. TECHNICAL CORRIGENDUM 3*, 2005.

ISO/IEC 15444-2:2004/Cor 4, *JPEG 2000 Image Coding System — Part 2: Extensions. TECHNICAL CORRIGENDUM 4*, 2007.

ISO/IEC 15444-2:2004/Amd 2, *JPEG 2000 Image Coding System — Part 2: Extensions. Extended capabilities marker segment*, 2006.

ISO/IEC 15444-4, *JPEG 2000 Image Coding System — Part 4: Conformance testing*, 2004.

ISO/IEC 15444-9, *Information technology — JPEG 2000 image coding system: Interactivity tools, APIs and protocols*, 2005.

ISO/IEC 15444-9:2005/Amd 1, *Information Technology – JPEG 2000 image coding system – Part 9: Interactivity tools, APIs and protocols. AMENDMENT 1: API's, metadata, and editing*, 2006.

ISO/IEC 15444-9:2005/Cor 1, *Information technology — JPEG 2000 image coding system: Interactivity tools, APIs and protocols. TECHNICAL CORRIGENDUM 1*, 2007.

ISO/IEC 15444-9:2005/Amd 2, *JPIP Extensions*, 2008.

ISO/IEC 15444-9:2005/Cor 2, *Information technology — JPEG 2000 image coding system: Interactivity tools, APIs and protocols. TECHNICAL CORRIGENDUM 2*, 2008.

ISO/IEC 15444-9:2005/Amd 3, *Information technology – JPEG 2000 image coding system: Interactivity tools, APIs and protocols. AMENDMENT 3: JPIP extensions*, 2009.

RFC 793, *Transmission Control Protocol*, 1981.

RFC 2045, *Multipurpose Internet Mail Extensions, (MIME) Part One: Format of Internet Message Bodies*, 1996.

RFC 2046, *Multipurpose Internet Mail Extensions, (MIME) Part Two: Media Types*, 1996.

RFC 2616, *Hypertext Transfer Protocol – HTTP/1.1*, 1999.

Non-Normative References:

The following documents are included for informational purposes only. The latest versions of these documents should be consulted.

STANAG 4545/AEDP-4, *NATO Secondary Imagery Format (NSIF)*

STANAG 7023/AEDP-9, *NATO Primary Image Format*

STANAG 4609/AEDP-8, *NATO Digital Motion Imagery Format*

Application for copies of ISO documents may be addressed to the respective national ISO representative.

Copies of NATO Standardization Agreements may be obtained from HQ NATO, Military Agency for Standardization, 1110 Brussels, Belgium, or from the www.nato.int website, if releasable to the general public. Some Standardization Agreements may only be released to NATO member nations.

3 Introduction

The JPIP protocol is defined in ISO/IEC 15444-9, Part 9 of the JPEG 2000 standard. It provides a communications syntax between a client and server that allows for the exchange of JPEG 2000 compressed imagery in a bandwidth efficient manner. JPIP uses the innate properties of the JPEG 2000 compression standard to deliver portions of the JPEG 2000 codestream between a client and server. With JPIP it is possible to browse a large image at reduced resolution and quality. When a client finds an area of interest in the image, additional resolution and quality can be requested to provide a high quality rendition of the area of interest. It is not uncommon to achieve this functionality while only transmitting 1-2% of the total file. For applications where the entire file must be transferred other protocols such as FTP (RFC 959) are more appropriate. JPIP is not meant as a replacement for these protocols.

JPIP allows for efficient client/server exchange of imagery data by providing mechanisms that allow servers to model the contents of a client's imagery cache. This minimizes the redundant transmission of data. Clients may also inform servers of their cache contents and manage the server's model of their cache. In circumstances where server modeling of a client's cache is not feasible, a client can explicitly request the exact portions of a JPEG 2000 codestream that it wants. These features allow not only efficient communication but also provide stop and resume functionality. A client/server exchange can be interrupted and resumed at a later time. If a client has saved its cache in between these sessions, it may inform the server of its cache contents and resume the data exchange.

The JPIP protocol is powerful and has many features. This RP makes recommendations regarding which features implementations should include. The JPIP standard allows a great deal of flexibility with regard to appropriate server and client responses to JPIP requests. This RP attempts to specify appropriate client and server behavior to promote JPIP interoperability and functionality without overly constricting client and server behavior.

4 Definitions

For the purposes of RP 0811, the definitions in the above normative references apply. The following additional definitions also apply.

Feature: Feature is a generic term used in this document to refer to JPIP capabilities that a server or client might implement. It is also used to refer to specific protocol options that a JPIP server or client might use. For example, *sessions* are a server feature (JPIP capability) that enables server-side cache modeling of a client's JPIP cache. *Independent headers* are a server feature (JPIP protocol option) that may be used to improve robustness to bit errors during JPIP exchanges. Features are implemented or enabled by a client or server supporting one or more JPIP functions.

Function: JPIP functions correspond to specific client/server responses and requests. These responses and requests are defined in ISO/IEC 15444-9.

Not Allowed: Any JPIP feature or function that is *not allowed* in this profile *shall not* be implemented by *any* compliant implementation.

Not Recommended: Any JPIP feature or function that is *not recommended* in this profile *may* be implemented by any implementation. Implementations are *encouraged to not implement* JPIP features and functions that are not recommended unless there is an operational need for the feature or function.

Optional: Any JPIP feature or function that is *optional* in this profile *may* be implemented by any implementation. It is left to each specific implementation as to whether or not the feature or function is implemented. Note that it is *never* allowed to incorrectly implement a JPIP feature or function.

Recommended: Any JPIP feature or function that is *recommended* in this profile *may* be implemented by any implementation. Implementations are *encouraged to implement* recommended features and functions but they are *not required* to implement them.

Required: Any JPIP feature or function that is *required* in this profile *shall* be implemented by all compliant implementations

5 Abbreviations and symbols

For the purposes of RP 0811, the definitions in the above normative references apply. The following additional abbreviations and symbols also apply.

LVSD: Large Volume Streaming Data. A NATO term that applies to systems that collect very large frame imagery on the order of 10^8 to 10^{10} pixels/frame. LVSD systems typically acquire imagery at 1 Hz or faster and may operate for hours at a time.

NSG: National System for Geospatial-Intelligence.

NIIA: NATO ISR Interoperability Architecture.

OGC: The Open Geospatial Consortium. Their website may be found at http://www.opengeospatial.org/. The OGC maintains standards related to geospatial and location services.

PJPIP: The Profile of JPIP defined by this RP.

WCS: Web Coverage Service. WCS defines a standard interface and operations that enable interoperable access to geospatial "coverages" (see http://www.opengeospatial.org/ogc/glossary/c). WCS is maintained by the OGC.

WFS: Web Feature Service. The OpenGIS Web Feature Service Interface Standard (WFS) defines an interface for specifying requests for retrieving geographic features

(http://www.opengeospatial.org/ogc/glossary/g) across the Web using platform-independent calls. The WFS standard defines interfaces and operations for data access and manipulation on a set of geographic features. WFS is maintained by the OGC.

6 Profile for JPEG 2000 Interactive Protocol (PJPIP)

The Profile for JPEG 2000 Interactive Protocol (JPIP) defines the syntax for a client to access whole or partial JPEG 2000 compressed imagery and imagery-related data residing on a JPIP server. This Recommended Practice is designed to tailor the options found within the ITU-T Rec. T.800 | ISO/IEC 15444-9:2004 - *JPEG 2000 image coding system: Interactivity tools, APIs and Protocols* for the NSG and NIIA architectures and their users.

It is recognized that JPIP metadata delivery procedure is a challenging topic within the NSG and NATO communities and will require input from numerous user groups in order to provide a final solution. This profile suggests a simple method of extracting metadata that can be used for any file format including NSIF (STANAG 4545) and NPIF (STANAG 7023). It is beyond the scope of this document to provide recommended practices for JPIP metadata delivery. A future metadata specific JPIP profile will be created to reflect a JPIP metadata approach approved by the community. At the community-wide level, a fundamental decision must be made whether the JPIP server is expected to handle the NSIF/NPIF/LVSD metadata, or whether this metadata should be requested and delivered via other means, for instance as part of the search-and-discovery methodology (e.g. WFS).

This Profile is compliant with ITU-T Rec. T.800 | ISO/IEC 15444-1:2004 - *Information technology – JPEG 2000 image coding system: Core coding system,* ITU-T Rec. T.800 | ISO/IEC 15444-2:2004 - *Information technology – JPEG 2000 image coding system: Extensions,* and ITU-T Rec. T.808 | ISO/IEC 15444-9:2004 - *JPEG 2000 image coding system: Interactivity tools, APIs and Protocols.*

6.1 JPIP Features

The JPIP protocol contains many different client and server requests and responses (i.e. JPIP functions, see Section 6.2). It is helpful to consider these functions in terms of the capabilities that they enable. There is not always a one-to-one correspondence between features and functions, some functions are critical for enabling multiple features. This section discusses various JPIP features and gives recommendations regarding their implementation.

6.1.1 Transport Type

ISO/IEC 15444-9:2005 defines two transport types for use with JPIP, “http” and “http-tcp.” The transport type is determined by the client indicating its desired transport type via the New Channel command (“`cnew`”, see ISO/IEC 15444-9 section C.3.3). The server responds to this request via the New Channel response header (“`JPIP-cnew`”, see ISO/IEC 15444-9 D.2.3). Figure 1 illustrates the relationship of the JPIP protocol to other transport protocols over which it may be carried. The JPIP standard defines the use of JPIP over strictly HTTP channels (“http”) and over HTTP with a dedicated TCP return channel (“http-tcp”).

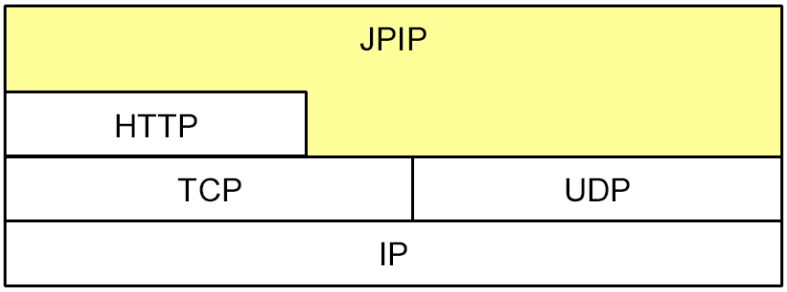

Figure 1 – Relationship of JPIP to other transport protocols

If the JPIP exchange is conducted using the HTTP protocol only (“http” transport), client requests and server response headers and status codes are sent over via the HTTP transport. Server response data (i.e. raw codestream, metadata-bins, header, precinct and tile data-bins) is also sent via the HTTP transport by tunneling the data over HTTP. If the “http-tcp” transport type is used then the client commands and server response headers are still sent via HTTP. Server response data, however, is sent via a dedicated TCP return channel.

Type	Recommendation		Notes
	Servers	Clients	
“http”	Required	Required	All JPIP implementations shall support HTTP only transport.
“http-tcp”	Recommended	Recommended	It is recommended that all JPIP implementations support HTTP-TCP transport.

Table 1 – JPIP Transport Type

Table 1 gives the PJPIP recommendations for JPIP transport type. All implementations shall support “http”. Oftentimes, network security concerns require that firewalls block the opening of additional TCP ports. Virtually all firewalls, however, allow HTTP traffic. The “http” mode allows JPIP exchange across firewalls that block opening additional TCP ports. The “http-tcp” transport type may be used to provide improved JPIP performance and finer control over bandwidth allocation. It allows servers to dedicate ports to a particular client. In situations where additional TCP ports are allowed on a network, “http-tcp” is the preferred transport type. It is recommended that JPIP implementations support this transport type. Note that the “http-tcp” transport type may only be used with sessions (see Section 6.1.7).

6.1.2 Image Return Type

The JPIP standard allows clients to request the type of JPIP streaming they desire. This is done by the client specifying an image return type. Based on the image return type that a client requests; some JPIP features may not be applicable or available. Clients request their desired stream media type(s) using Image Return Type command (“`type`”, see ISO/IEC 15444-9 C.7.3). Servers respond to this request using the “`JPIP-type`” response (see ISO/IEC 15444-9

D.2.17). MIME types defined within RFC 2046, “Multipurpose Internet Mail Extensions, (MIME) Part Two: Media Types” are allowed by ISO/IEC 15444-9 with the intent that the JPIP server will provide an appropriate transcoding of the media so that it may be delivered via JPIP. PJPIP does *not allow* the use of RFC 2046 mime types as valid image return types.

The JPIP standard defines three additional image return types beyond those found in RFC 2046. These types are “jpp-stream” (JPP-stream), “jpt-stream” (JPT-stream) and “raw” (RAW). Choosing between these three image return types profoundly affects the nature of a JPIP exchange and the client/server capabilities available within that exchange. Of the three image return types, the RAW type is the simplest to understand and provides the least capability. JPIP exchanges that use the RAW image return type simply send the image codestream from server to client. The JPIP server is to simply send the sequence of bytes in the codestream to the client unchanged. In this manner it is very similar to using FTP to send an image to the client. In ISO/IEC 15444-9, this manner of operation is referred to as “raw codestream” mode.

The JPP-stream and JPT-stream image return types provide much more flexibility. The JPIP standard refers to these modes of operation as “incremental codestream” operation. The JPT-stream mode transfers data from the JPIP server to the client by incrementally serving the tile-part codestreams. A tile-part codestream may be thought of as taking all of the tile-parts for a given tile and concatenating them in order. The JPIP server then incrementally delivers this data (tile data-bins) taking into account the client’s view-window, layer, quality and byte range requests. Depending upon the progression order of the codestream a server may be required to deliver additional codestream information to honor a client’s request.

Clients can take the codestream data received from a JPT-stream exchange and concatenate it to form a valid JPEG 2000 codestream that any JPEG 2000 decoder can interpret. Some minor processing is typically needed to adjust or insert some tile-part marker segments or alter the codestream for data that was not received. This might happen if the client only viewed a portion of the image or did not view all resolution levels or quality layers. In this case the server might not have delivered these portions of the codestream and some modifications will be necessary.

The JPP-stream image return type is the most powerful and complex of the three return types. For this return type the server incrementally delivers precinct data-bins to the client. To better understand precincts, consider Figure 2. The left hand side of the figure shows an image; its wavelet decomposition is on the right hand side, and has been scaled so that it may be viewed as an image. Overlaying the wavelet transformed image are graphics that indicate subbands, code blocks and precincts.

The red lines in Figure 2 denote the wavelet subband boundaries and the green lines indicate the code block boundaries within the subbands. The code block represents the smallest decodable unit within JPEG 2000. To decode a spatial region within a JPEG 2000 compressed image, a decoder must decode the code blocks throughout the wavelet transform that correspond to the spatial region. Also indicated in the figure are precincts. Information about the code blocks in a JPEG 2000 codestream such as, the number of bitplanes each code block contributes to each quality layer, are encoded in structures called packet headers. It is possible to create a single packet header that includes all code blocks in a subband or we may create several smaller packet headers.

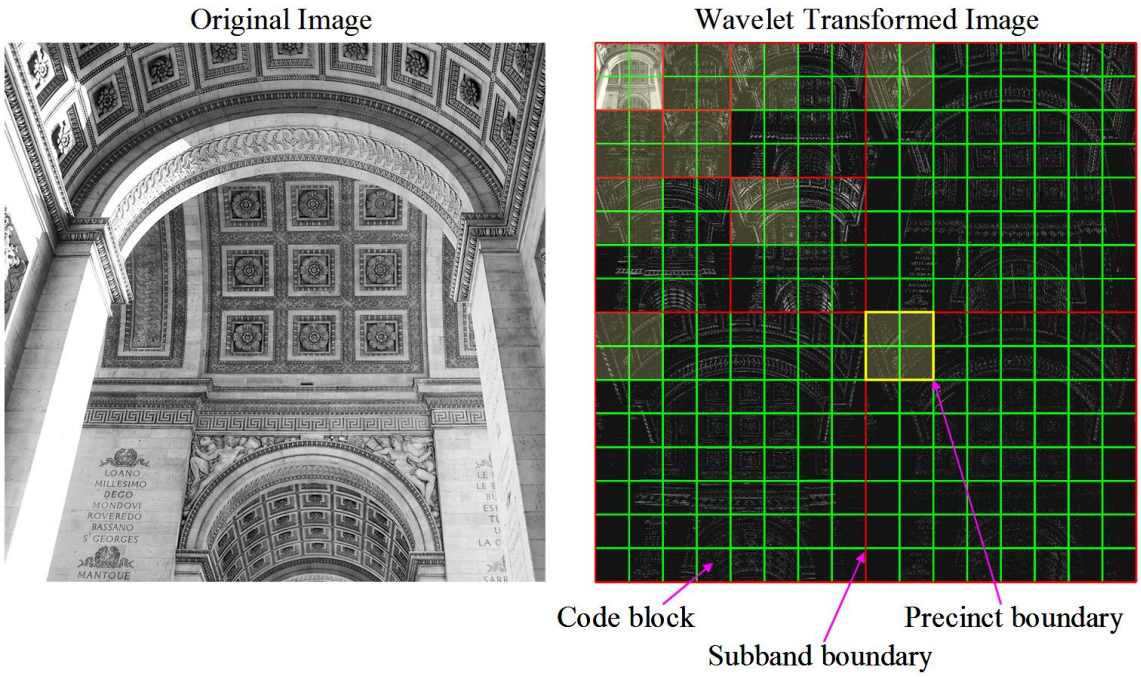

Figure 2 – Subbands, code blocks and precincts

To provide improved spatial access within a JPEG 2000 codestream, it is best to use smaller packet headers. If the packet header information for all code blocks within a subband is encoded together and we wish to only decode a small spatial region in the image, the decoder will have to decode much more packet header information than needed. For large images, the amount of information within the packet headers can be significant. If the image is being streamed over a low bandwidth link, large packet headers can slow down the delivery of a spatial region of interest within JPIP. It is therefore good practice to break up these large packet headers into smaller pieces.

Precincts are the mechanism used to do this. The yellow box and yellow-shaded areas in Figure 2 indicate precincts. It is good practice to use precincts that comprise a 2x2 array of code blocks; this is indicated in the figure. If the original file that the JPIP server is delivering did not make use of precincts (i.e. it has one precinct per subband), the server should "re-precinct" the codestream. Adding/changing the precinct size in a JPEG 2000 codestream is relatively easy and it is recommended that all JPIP servers do this especially for very large imagery.

When a JPIP server delivers data using the JPP-stream image return mode, it sends portions of the codestream based on the precinct structure. For example, in Figure 2, if a client wishes to receive the top left corner of the image, the JPIP server would send the codestream data associated with the yellow-shaded precincts. A unique index is sent with each precinct that tells the client which tile, component and precinct within the tile-component the data belongs to. This mechanism allows a great deal of flexibility in how the data is delivered by the JPIP server.

The JPP-stream precinct data-bins form a self-describing codestream format. With the precinct data-bin mechanism, it is possible to deliver JPEG 2000 codestreams in a fashion that cannot be easily duplicated using the JPEG 2000 raw codestream format. The JPP-stream delivery

mechanism allows an arbitrary delivery order of precinct and code block data. Additionally, many of the JPEG 2000 marker segments that tell a decoder how the data within a JPEG 2000 file is structured are no longer needed with precinct data-bins.

PJPIP *requires* that all JPIP server and client implementations support the JPP-stream image return type. Although the JPP-stream image return type is more complex to implement, it is extremely powerful and offers the type of interactive client/server capabilities for which JPIP was created. This profile *recommends* that all JPIP server implementations support the JPT-stream image return type. Support of JPT-stream image return type is *optional* for clients. The RAW codestream image return type is *optional* for both servers and clients. Servers are encouraged to implement this image return type since it is a very simple delivery mode. Table 2 summarizes the PJPIP recommendations for the three JPIP image return types and RFC 2046 mime types.

Type	Recommendation		Notes
	Servers	Clients	
"raw"	Optional	Optional	Raw codestream image return type is optional for both servers and clients.
"jpt-stream"	Recommended	Optional	Servers are recommended to implement JPT-stream image return type. JPT-stream is optional for clients.
"jpp-stream"	Required	Required	Servers and clients shall implement the JPT-stream image return type.
Mime types* (RFC 2046)	Not Allowed	Not Allowed	Non JPEG 2000 image return types are not currently supported by PJPIP.

*Note: It is anticipated that future versions of this profile will allow some RFC 2046 mime types. For example, transcoding and JPIP delivery of JPEG compressed (ISO/IEC 10918-1) imagery may be added.

Table 2 – JPIP Image Return Type

6.1.3 JPIP Message and Data-bin Options

JPIP defines four types of data-bins; metadata, main header, precinct and tile. Implementations must of course support the types of data-bins associated with their chosen image return types. In addition to the types of data-bins there are options concerning their delivery that an implementation may choose to utilize. All metadata and imagery data in a JPIP exchange are transferred via the four data-bin types. In a typical JPIP exchange, multiple data-bins are exchanged between the server and client.

Data-bins are typically sent as a series of messages. This is done for two primary reasons. First, breaking the data-bins into smaller messages prevents the server from overloading the communication channel when sending large amounts of data all at once. Second, JPIP allows the client to pre-empt the server's delivery of imagery data (depending upon server control fields). A

server may pre-empt its current delivery of imagery data whenever the client sends a new view-window request before the server has completely finished servicing the previous request. This would not be possible unless data-bins are broken into a series of messages. Breaking data-bins into a series of messages provides an interactive JPIP server/client experience.

Messages consist of a header and a body. The header contains information that defines the data-bin to which the message belongs and the body contains metadata or imagery data. In additional to the header and body, a JPIP server will insert End of Response (EOR) codes that tell a client the server has stopped responding to the client's request. There are several reasons why a server might send an EOR code. If the server has sent all imagery data needed to fulfill a client's view-window request, the server will send a code indicating this. Servers will also send EOR codes indicating that the current message is finished when a byte limit is reached. For example, a client may request a server to limit its message lengths to 2000 bytes. When the server reaches this limit, the server stops sending data and includes an EOR code of four (byte limit reached) to the client. To receive further imagery data, the client simply repeats its view-window request and the server will continue sending data from the point at which it had stopped (assuming server-side cache models are in use). The various EOR codes are given in Table D.2 of ISO/IEC 15444-9.

The message headers can take one of two forms: an independent form and a dependent form. The first message sent for any given data-bin must be of the independent form, since this type of header identifies the type of data-bin being sent and the codestream number associated with it. Subsequent messages for the same data-bin may use the dependent header form, which is slightly shorter. If the dependent header form is used, the client is to assume that the current message belongs to the data-bin previously indentified by the last independent header. Clearly, usage of the dependent header form requires that the client receives JPIP messages in their proper order. For reliable communications this assumption is valid.

PJPIP *recommends* that all JPIP servers use the independent header form at all times. Furthermore, this profile *requires* that all clients properly interpret the dependent message header format. Servers may choose to utilize the dependent message header format if they are certain of reliable communications channels. Clients shall always be able to interpret the shorter dependent header form in case the server they are connecting with chooses to use it.

Certain JPIP data-bins have two forms: their normal form and an "extended" form. The precinct (JPP-stream) and tile (JPT-stream) data-bins have an extended form where additional data regarding the amount of the codestream received is communicated. For the precinct data-bins the extended form indicates the number of completed quality layers. The extended form of the tile data-bin includes the number of resolution levels completed. In both cases, this information is not needed by the client, but it may be useful.

Clients request usage of the extended headers by the server via the image return type request (`"type"`, see ISO/IEC 15444-9 C.7.3). To request extended precinct data-bins from a server the client appends a `";ptype=ext"` to the request. For extended tile data-bins the client appends `";ttype=ext"` to the image return type request. See Table D.2 in ISO/IEC 15444-9 for details. PJPIP *requires* all JPIP servers to implement the extended data-bin forms. Usage of extended data-bins is *optional* for JPIP clients conforming to this profile. Servers *shall not* return

extended data-bins unless a client explicitly requests them.

Option	Recommendation		Notes
	Servers	Clients	
Independent message headers	Recommended	Required	Independent message headers shall be used in the first message when sending a data-bin. It is recommended for servers to use independent headers for the remaining messages for the data-bin. Clients shall be able to interpret independent message headers.
Dependent message headers	Optional	Required	Servers may choose to use the dependent message header form for messages that are not the first ones associated with a data-bin. Clients shall be able to interpret dependent message headers.
Extended data-bins	Required	Optional	Servers shall provide extended data-bin types whenever a client requests them. Clients may choose to request the extended data-bin types.

Table 3 – Message and Data-bin Options

Table 3 summarizes the PJPIP recommendations regarding independent and dependent message headers and extended data-bin types.

6.1.4 Metadata Delivery

PJPIP does *not recommend* that metadata by sent via JPIP.[1] A future recommended practice will provide guidance regarding the delivery of metadata when using JPIP. If implementations choose to send metadata over a JPIP exchange, this section provides guidelines that should be followed and points out some potential pitfalls. It is important to make some distinctions between metadata and metadata-bins. This Recommended Practice does not recommend that metadata-bins be avoided. Metadata-bins are an essential part of the JPIP protocol. PJIP recommends that large amounts of metadata not be carried via metadata-bins. For example, raw codestreams (i.e. RAW image return type) are delivered within a metadata-bin.

To understand some of the issues, consider a JP2 or JPX file that contains a number of XML or UUID boxes that contain metadata (see ISO/IEC 15444-1 and 15444-2 for information regarding the JP2 and JPX formats and box structures). By default all boxes within a JP2 or JPX file,

[1] Note that ISO SC29/WG1 is currently drafting amendments to ISO/IEC 15444-2 and ISO/IEC 15444-9 that could potentially improve the delivery of metadata via JPIP. A future version of this RP (or another RP covering LVSD metadata delivery) may make use of theses changes should they be found useful for LVSD systems.

except contiguous codestream boxes, belong to metadata-bin zero. JPIP servers deliver this metadata-bin prior to delivery of an incremental codestream (for raw codestreams the entire codestream is served out of a metadata-bin). If the amount of metadata within a file is significant and the channel bandwidth is low, this may lead to an unacceptable delay in transmission of the imagery data.

Placeholder boxes (see Annex A in ISO/IEC 15444-9) may be used to "move" XML or UUID boxes out of metadata-bin zero into other metadata-bins. The JPIP server then transmits the placeholders and the client then specifically requests the other metadata-bins using a "metareq" request (see Annex C.5.2 in ISO/IEC 15444-9). One can imagine a hierarchical breakdown of the various metadata-bins by the server that the client can request. The "metareq" request allows for retrieval of box contents based on metadata-bin number, placement within the hierarchy, box type, etc. The "metareq" request can be used to retrieve an entire box, box headers only or the box headers with a portion of the box. This last feature could be used for example to pull the UUID out of a UUID box by requesting the box header along with the first 16 bytes of the box which contain the UUID.

One can imagine that a hierarchical breakdown of the metadata boxes could be devised and then the client could request boxes containing only the desired metadata. The above technique, however, is only applicable to JPEG 2000 file formats. Sending an NSIF or NITF file over JPIP using this technique will not work. We might recast the NITF file into a JP2 or JPX file and transfer all of the file, image and text segments, DESs, etc. and their metadata into various boxes. Then, this new JP2 or JPX file could be sent as described above. Alternatively, we may instead use raw codestream delivery and treat the entire NSIF file as a block of bytes. The client may then request explicit byte ranges out of the file for delivery to retrieve desired pieces of metadata. To incrementally deliver any imagery in the file will require a second JPIP channel since the incremental codestream delivery and raw byte delivery cannot be used on the same channel.

The issue with the above schemes is that they require a standardization of the JPIP server's behavior that lies outside of the scope of ISO/IEC 15444-9. Finding commercial JPIP implementations that would support such a customization might prove difficult. Besides, metadata delivery issues have already been addressed through other standards such as WFS from the OGC. The geospatial community has developed their own sets of standards to deal with querying and retrieving metadata. Their techniques are widely supported and more elegant than those described here. For this reason PJPIP does *not recommend* the transfer of significant amounts of metadata via JPIP.

6.1.5 Other Codestream Types

ISO/IEC 15444-3 describes the Motion JPEG 2000 format. Motion JPEG 2000 is encoded as a sequence of JPEG 2000 codestreams; no temporal coding processes are used. JPIP is capable of delivering Motion JPEG 2000 files, but there is *no requirement* for PJPIP compliant servers and clients to support Motion JPEG 2000.

ISO/IEC 15444-2 describes extensions to the baseline JPEG 2000 encoding processes. One of these extensions is the Multiple Component Transform (MCT) framework. The MCT framework provides mechanisms for the compression of multispectral (MSI) and hyperspectral (HSI) imagery. It is anticipated that LVSD platforms will eventually incorporate spectral imagers.

There is currently *no requirement* for PJPIP servers and clients to support delivery of JPEG 2000 files that make use of the MCT framework. It is anticipated that at some point in the future this will become a requirement. Currently ISO has added one new client request field to provide partial support for JPEG 2000/MCT files (see "`mctres`" in ISO/IEC 15444-9:2005/FPDAM 3). Future addendums or editions of the JPIP standard may add increased support MSI/HSI JPEG 2000 compressed imagery. When it becomes necessary for PJPIP servers and clients to support MSI/HSI JPEG 2000 compressed imagery, a new edition of this RP will be issued. It is not anticipated that all PJPIP servers and clients will be required to support spectral imagery, but LVSD systems will most likely be required to do so.

ISO/IEC 15444-6 defines the JPEG 2000 Compound Image File Format. This standard contains the JPM file format which defines how to use of JPEG 2000 (as well as MMR, JBIG) in the context of compound documents. ISO/IEC 15444-9:2005/ Amd 1:2006 contains extensions to the JPIP standard to support JPM files. There is *no requirement* for PJPIP compliant servers and clients to support the JPM file format or any JPIP extensions related to that format.

6.1.6 Server Transcoding

The logical target served by a PJPIP server need not physically exist. As described in Annex C.2.1 of the JPIP standard, the logical target may represent either a physical file or object or it may be created virtually by the server on request. Furthermore, the logical target may be transcoded from the original JPEG 2000 codestream form. PJPIP servers may optionally transcode the requested JPEG 2000 codestream target into an alternate format, so long as the reconstructed pixels within the region of interest are equivalent to those that would have been received by the client had the original image been served directly. We have already encountered one scenario where transcoding is desired: to "re-precinct" a JPEG 2000 codestream. In particular, the following transcoding operations are permitted:

- Introduction of smaller precincts. For JPP-stream image return type, a server may choose to "re-precinct" the codestream. New precinct dimensions *shall be no smaller* than twice the code-block width by twice the code-block height unless subband dimensions prevent this (at reduced resolutions subband size might be smaller than a codeblock). The new precincts *shall* be an integral number of codeblocks in width and height. Smaller precincts allow finer granularity than tile boundaries in responding to a requested region of interest. PJPIP *recommends* that servers re-precinct codestreams to provide improved spatial region of interest access. Changing precincts to correspond to a 2x2 group of code-blocks is a good choice.

  Transcoding from maximal precincts to a smaller precinct size requires only a change to the COD and/or COC marker segments and a re-ordering of packets (with new packet headers). There is no need to perform entropy decoding/encoding or a wavelet transform as part of this transcoding step. This may be accomplished with relatively little computation on the server side (especially if PLT marker segments are available), and can easily be undone on the client side if it is necessary to save a file using maximal precincts. Requiring that the precinct dimensions be an integral number of codeblocks prevents modifications to codeblock partitions (because codeblock sizes are constrained by precinct sizes) and avoids entropy decoding/encoding and reverse/forward wavelet

transform. Servers may trade off the benefits of lower network bandwidth versus the extra processing overhead of performing this transcoding step.

- Introduction or removal of metadata boxes. JPIP was designed to deliver image data. Since metadata is not essential to the reconstruction of image data, JPIP servers may remove metadata at their discretion without impacting image delivery. If an implementation decides to use JPIP for metadata delivery the removal of unnecessary metadata boxes and the introduction of placeholder boxes to break metadata up into a manageable hierarchy are *recommended*. If an implementation decides to use alternative metadata delivery mechanisms outside of JPIP, then metadata boxes should be removed and the contents delivery by the alternate delivery mechanism. In either case, modification of the metadata box structure in a JP2 or JPX file is likely to occur.

- Removal of COM (comment) marker segments. Since comments are not required to properly interpret the image data, they can be omitted to reduce the network bandwidth usage. PJPIP makes removal of COM marker segments *optional* for servers.

- Removal of TLM (tile-part lengths, main header) marker segments. Since TLM marker segments don't serve a purpose when managing an image as precincts in cache (JPP-stream) rather than from a file, they can be omitted to reduce the network bandwidth usage. The original tile-part lengths will not typically correspond to the data in the client's cache, unless the full image has been delivered and no transcoding was performed on the original image. Clients should typically generate a new TLM marker segment if saving the contents of its cache to file. PJPIP *recommends* the removal of TLM marker segments for JPP-stream image return types.

- Removal of PLT (packet lengths, tile-part header) marker segments. PLT marker segments serve no purpose when managing an image as precincts in cache (JPP-stream) rather than from a file, they can be omitted to reduce the network bandwidth usage. Besides, the original packet lengths will not typically correspond to the data in the client's cache, unless the full image has been delivered and no transcoding was performed on the original image. Clients should typically generate new PLT marker segments if saving the contents of its cache to file. PJPIP *recommends* the removal of PLT marker segments for JPP-stream image return types.

Another way of looking at these last three bulleted items is that the COM, TLM, and PLT marker segments are optional in the main header and tile header data-bins. In fact, for JPP-streams, if the tile-part headers contain only SOT, COM, PLT, and/or SOD marker segments (as is typically the case), the tile header data-bins may have zero-length bodies.

6.1.7 Sessions

The JPIP protocol may operate as either a stateless interchange of sequential messages and data transfers, or as a session which maintains state from one request to the next. Sessions retain client capabilities and preferences supplied in previous requests, and typically include a log of any information the server previously sent to the client so that the server need not re-transmit such data in response to subsequent requests.

PJPIP *requires* the JPIP server to support both stateless requests and sessions. This allows the JPIP client to choose the appropriate type of request for a specific application. In general, sessions require smaller, less complex requests with less redundant response data from the server. Interactive viewing clients should typically make requests within a session to optimize network bandwidth and thus improve overall response latency. On the other hand, stateless communication requires fewer burdens on the server to maintain state information between requests and may offer benefits in a distributed environment where a single target image may reside on multiple server hosts. The client cache management requests associated with stateless communication also allows for suspend and resume functionality even if a session has been initiated.

When delivering imagery via a JPP-stream or JPT-stream within a session, the server typically maintains a cache model of what is in client memory. This allows the server to avoid re-transmitting redundant data to the client for more efficient communication. Although redundant data is superfluous and may be ignored by the client, it can drastically impact network bandwidth and overall system latency. Thus, while maintaining a cache model is optional from the standpoint of delivering correct data, it is highly encouraged and expected of PJPIP servers.

If the client discards any data from its cache (e.g., to avoid excessive memory use with very large images), it signals those subtractions to the server, and the server is *required* to adjust its cache model of the client accordingly, such that missing data will be retransmitted as necessary upon subsequent requests. The client may also signal additions to the cache model, and it is *recommended* the server adjust its cache model so as to avoid sending redundant data that the client already has. For instance, additive cache model adjustments may be made if the client starts a new session but already has some data in cache from a previous session. Furthermore, when dealing with multiple codestreams within a target (e.g., with motion imagery), the concept of a cache model-set ("mset") may be used to signal wholesale deletions of codestream indexes from the client's cache.

A client may open more than one channel within a single session. Multiple channels allow clients to issue simultaneous requests for different regions of interest to be served in parallel. In contrast, if multiple requests are made on a single channel, requests are handled serially with each new request canceling any previous request in the queue (unless it specifies to wait for the previous request to complete, see "`wait`" in ISO/IEC 15444-9, Annex C.7.2). PJPIP servers are *required* to accept at least the first four (TBD) new-channel requests for a single target within a session. If more channels are requested by a client, the server has the option to support or reject such requests. Although permitted, PJPIP servers are not required to honor new-channel requests for different targets within a single session. Multiple channels within the same session that reference the same logical target share a common cache model. Thus redundant data need not be sent, and the client will need to combine the data received over the various channels. Please refer to Annex B of ISO/IEC 15444-9 for further details on sessions, channels, and cache model management.

6.1.8 Image Upload

JPIP supports image uploading from the client to the server. Clients make "`upload`" (see ISO/IEC 15444-9 Annex C.9.1) requests to the server to send new portions of the JPEG 2000 codestream back to the server with the intent that the server will use along with the original

codestream to create a new logical target. Clients may use the RAW, JPT-stream and JPP-stream image return types for their update request. The behavior of the server in these three cases is detailed in Annex E of ISO/IEC 15444-9.

PJPIP makes *no requirements* on clients or servers to support the JPIP image upload feature. If an implementation wishes to make use of this feature it shall obey the policies set forth in Annex E.4.1 of ISO/IEC 15444-9. If a PJPIP server receives an "`upload`" request and the server does not support image upload, the server *shall* respond an HTTP 501 (not implemented) response (see ISO/IEC 15444-9, Annex E.3.7). If the server accepts the request it *shall* respond with either a 201 (created) or 202 (accepted) response (see ISO/IEC 15444-9, Annex E.3.2 and Annex E.3.3). If a server does not support the particular type of image upload (RAW, JPT-stream or JPP-stream) it *shall* respond with a 415 (unsupported media type) code (see ISO/IEC 15444-9, Annex E.3.6).

It is *recommended* by PJPIP that clients always use a target ID of 0 or a new URL for the first image upload/update. If the server accepts the request, it *shall* respond with a new target ID and all further image uploads *shall* be directed at that target ID. This is a relatively simple way to lock target IDs and prevent simultaneous image uploads/updates from multiple clients. It remains the server's responsibility to maintain the various versions of the codestream while it is being accessed. After all image upload sessions have completed, it is the implementation's responsibility to decide how to merge the various versions of the codestream if desired.

6.1.9 Indexing

Annex I of ISO/IEC 15444-9 defines a set of JP2/JPX boxes that may be used to "index" a JPEG 2000 file. Boxes are defined that give pointers to the main header marker segments, tile-parts, tile-part header marker segments, packets, packet headers, codestream fragments and other constructs within a JP2 or JPX file. The intent of these index boxes is to provide quick server and/or client side parsing of the file to locate desired portions of a codestream or metadata contained within the file. To better understand the application of these indices we consider two cases, server ingest and delivery of a JPEG 2000 codestream over JPIP and clients requesting RAW codestream delivery.

The index boxes defined in Annex I could be generated by an image library whenever a JPEG 2000 compressed file is first ingested and stored. When clients subsequently request JPT-stream or JPP-stream delivery of the file from a JPIP server, the index information contained within the file will aid the JPIP server in locating portions of the codestream for delivery to the client. Remember that the JPIP server need not pass these index boxes on to the client. The JPIP server can use this information and deliver a codestream equivalent where the index boxes are removed. From the client's perspective, it would be unaware of all of the additional index metadata. Alternatively, if a client requests the same file to be delivered in a RAW codestream format, the index boxes will provide valuable information to the client for making subsequent byte range requests. In this scenario the JPIP server can pass along all of the index information to the client so that the client can control the JPIP exchange and explicitly request the bytes out of the codestream that it wants.

PJPIP makes *no recommendation* regarding the use of the index boxes described in Annex I of ISO/IEC 15444-9. There is no mechanism for a client to request that a server index a JPEG 2000

file. If a server implementation chooses to support indexing, PJPIP *recommends* that the index boxes only be delivered to clients that make RAW codestream image return type requests. If a client requests JPT-stream or JPP-stream image return types, PJPIP *recommends* that the server pulls the boxes and delivers an equivalent representation. The server may utilize the index boxes to aid in its JPIP codestream delivery, but the index boxes should not be passed on to the client.

6.2 JPIP Functions

JPIP functions are comprised of client requests and server responses. For a detailed description of all client request fields, please refer to Annex C of ISO/IEC 15444-9. Client requests consist of a series of fields and subfields that identify the logical target and requested data associated with the view-window. Section 6.2.1 describes the PJPIP restrictions and conditions for these fields. PJPIP compliant servers must respond to every client request using the reply syntax described in Section 6.2.2, whether or not any actual codestream data is included in the response. Section 6.2.3 highlights the components of the server's response, including any specific issues relative to PJPIP. For a detailed description of the server response signaling, please refer to Annex D of ISO/IEC 15444-9.

6.2.1 Client Request Fields

Table 4 describes the fields used by the JPIP client to request data from the JPIP server. The "Notes" column is not intended as a complete restatement of the contents of Annex C but rather as additional information for JPIP implementers, including any particular requirements imposed by this profile.

Note that the PJPIP requirements are described from the perspective of what the server must be prepared to handle. Just because a server is required to handle a request field does not mean that the client must include that request field in every request. Rather, this table defines a profile of available features from which the client may choose when assembling requests.

The key to the server support requirement level is as follows:

- 0: No support required – the server is not required to understand the syntax and may issue an error if the field is requested, using either status code 400 if the field is unrecognized or 501 if the server recognizes the field as a valid request that it does not implement.
- 1: No functionality required – the server may ignore the requested field, but it must parse the field successfully without issuing an error and may be required to issue an appropriate response.
- 2: Validation required – the server need not implement multiple options, but it must verify that the request does not include any unimplemented requirements and issue an error with status code 501 if the request cannot be satisfied.
- 3: Limited or conditional functionality required – the server must implement some of the options as described in this table, perhaps only under certain conditions; depending on the field, some unimplemented options may be ignored while others require an error with status code 501.
- 4: Full functionality required – the server must implement all options of the requested field as described in Annex C of ISO/IEC 15444-9.

JPIP Request Field	Server Support Required	Notes
Target Identification Request Fields		
target	4	It is not required that the original named resource refer to a physical file stored on disk. It is conceivable, for instance, that images could be stored in a database where individual images have unique resource identifiers, in which case the logical target (from the client's perspective) could be an individual codestream. At the other extreme, many physical files (each with an individual raw codestream) could be treated by the server as a single logical target with many codestream indexes. If the target request field is missing and the request is made over HTTP, the original named resource is specified through the path component of the JPIP request URL.
subtarget	3	Servers shall support the sub-target field in conjunction with the return type "raw" in reference to a byte range relative to the start of the logical target. For instance, the sub-target could be used to parse through all portions of a NSIF file that are not a part of the JPEG 2000 code stream. The server is not required to support the sub-target for image return types other than "raw".
tid	4	The server is required to provide a unique target ID for each target. If the logical target changes, the server must provide a new target ID for future requests. Furthermore, any request using the outdated tid will result in the server providing the client with status message 404. The server also has the option of retiring a target ID even if the logical target has not changed.
Session and Channel Request Fields		
cid	4	Specifies the channel ID to associate the request with an existing channel within a session. Once a session has been started, subsequent requests typically will use only the channel ID instead of the target, sub-target, and target ID. This may be used in conjunction with the "cnew" field to request a new channel on the same session as that used by the identified channel ID (see details below).
cnew	4	This field is used to request a new channel within an existing or new session. If a channel ID is present, the request is for a new channel on an existing session. Multiple channels on a session for a single target may be useful to allow parallel requests for different windows of interest using a shared cache model. PJPIP servers are required to support at least four (TBR) channels on each session – if a client requests more than this, the server may (at its discretion) disallow the new channel and instead deliver the data on the existing channel – in this case, the server would not issue a new channel response header. Although legal within the JPIP standard, PJPIP servers are not required to support more than one target per session. Since separate targets have their own cache model, there is little or no benefit to supporting multiple targets per session. Thus, if the client requests a new channel within an existing session that identifies a new target, and if the server is unwilling to handle this, it may return an error code. Clients should typically plan to use separate sessions for each target.
cclose	4	Closes one or more channels belonging to a single session.
qid	4	For the reliable HTTP and the HTTP-TCP transports, the server may assume that the requests have been received in the order issued and that the client has issued requests with valid (non-decreasing) ID values, and thus

JPIP Request Field	Server Support Required	Notes
		no special handling is required. If a client issues an ID the server must issue a "JPIP-qid" response using this ID value. Servers must honor request ID order for all requests that include them. Requests that do not contain a request ID are handled first come first served.
View-Window Request Fields		
fsiz	4	Specifies the frame size, used to identify the resolution associated with the requested view-window. The server shall implement all options, including round up, round down, and round to closest. If not present, the main header and metadata are requested without any image data.
roff	4	Specifies the offset of the region of interest. If not present, the offset defaults to 0.
rsiz	4	Specifies the size of the region of interest. If not present, the server shall deliver no image data, but may or may not deliver the main header and metadata (at the server's discretion). Because of this possible ambiguity, if a client desires the main header and metadata, it should omit both the frame size and region size, in which case the specification requires the server to consider the main header and metadata to be requested. It is illegal to specify a region size without a frame size.
comps	4	Specifies the components (spectral bands) to include in the window. If not present, all available components are included.
stream	4	Specifies the codestream indexes to include in the window. If not specified, the default codestream index of 0 refers to the first JPEG 2000 image segment in a file. The stream field may optionally be used to support real-time collects by identifying a range of indexes that refer to codestreams within the logical target that will become available at a future time. Any region of interest is applied to all codestream indices within the request, and as each codestream becomes available the data for the specified region will be streamed directly to the client. For instance, a client could request "stream=0-", indicating all codestreams, starting with index 0 and with no upper bound. A specialized server which is handling a live data collection (or any case where additional codestreams are dynamically added to a logical target) could interpret this as a request that includes future data to be delivered as it becomes available. With this model, as each new codestream is added to the target, if its codestream index is included in the specified range, the requested region from this new codestream would be "pushed" to the client, as long as the request has not been terminated or timed out. A standard server that does not support a dynamically growing target would simply deliver the region of interest from those codestream indexes that are requested and exist in the logical target.
context	2	This field is typically used with higher level image targets to indicate compositing layers within a JPX file or video tracks within an MJ2 file. For a simple JP2 file with a single codestream, the client may request "context=jpxl<0>" to refer to that single codestream, and the server should accept this. PJPIP servers are not required to handle compositing or video tracks, the server may issue an error status 501 (unimplemented) if any context other than "jpxl<0>" is requested and the server is unable to process the specified context. The context field has no meaning for raw codestreams and thus may be interpreted by the server as a non-existent compositing layer (delivering no imagery) or as an error condition – in

JPIP Request Field	Server Support Required	Notes
		general, clients should not specify a context when requesting data from a raw codestream.
srate	4	This field is typically used in conjunction with motion imagery that contains timing information such as MJ2 files. It may also be used for a timed sequence delivery of codestreams. This functionality is useful for LVSD systems where a sequence of still frames is collected at some number of frames/second. If timing information is available for the frames (through ancillary metadata or information regarding the collection such as file time stamps), the "srate" request instructs the server to temporally subsample the frame delivery to match the requested rate. If the files do not have source timing information and the "srate"request field is present, the server is to assume that the frames have source times that are separated by the reciprocal of the "srate"field. Clients may use the "srate" request in conjunction with the "drate" request field. In this case the "srate" field sets the frame source times and the "drate" request filed sets the desired delivery rate to the client (note: "drate" sets a multiplicative factor).
roi	1	Specialized servers may implement named regions of interest, but this specification does not identify any standard names that general servers are expected to recognize. As prescribed in Section C ISO/IEC 15444-9, the server shall ignore this field if it does not know how to handle it and instead use the region size ("rsiz") and offset ("roff") if specified and respond with "JPIP-roi: roi=no-roi".
layers	4	Specifies the number of quality layers that belong to the view-window request. The default value of 0 indicates that all quality layers are of interest. A client that knows the profile used to compress an image may use this field to effectively specify a bit rate by requesting the corresponding number of quality layers.
Metadata Request Fields		
metareq	0	This field may be used with JP2 formatted imagery to access the contents of the metadata boxes. This field is irrelevant for raw JPEG 2000 codestreams. It applies only to JP2 and JPX file formats and PJPIP servers are not required to support this function. It is envisioned that JPIP metadata will be handled via a different mechanism, rather than trying to map it into JP2 boxes as would be necessary in order to use this metadata request field. Although the server need not handle the explicit metadata request via the "metareq" field, it must deliver any implicit metadata request as described in ISO/IEC 15444-9. In particular, for raw JPEG 2000 codestreams, the server is expected to explicitly signal a metadata bin 0 with a zero-length body to indicate that there is no metadata available.
Data Limiting Request Fields		
len	4	Specifies a limit on the number of bytes of response data (excluding response headers). The client may specify "len=0" to request only response headers with no response data.
quality	1	This optional field may be used with some servers to specify a quality level between 0 (lowest) and 100 (highest). PJPIP does not specify a policy for interpreting the quality values, and thus the servers are not required to support this function. If the server does not support the quality

JPIP Request Field	Server Support Required	Notes
		specification, it shall respond with a value of "-1", and no error shall be generated. A specialized server may try to use this field to approximate the compression quality factor commonly used in JPEG-DCT, but there is no standard defined manner for doing so.
Server Control Request Fields		
align	2	The server is not required to support alignment on natural boundaries. At a minimum, though, the server must be able to parse this field and respond without error if "align=no" is specified. If the request is for "align=yes" but the server is unable to align the data on natural boundaries, it should issue error status code 501 (not implemented). A potential alternate server implementation would be to always align data on natural boundaries, regardless of requested alignment; i.e. it is not an error to provide aligned data even if the client does not request it.
wait	4	If specified, the server shall complete the response to the previous request on the current channel before responding to this request. By default, previous requests are terminated when a new request is received, which is useful for interactive applications, where there is no need to deliver data to an old view request once the region of interest has changed.
type	3	Specifies the type of response data requested. The server is required to support the "jpp-stream" and "raw" types, but is not required to support the "jpt-stream" type. Clients typically should use the "jpp-stream" type for interactive delivery of windows of interest, but may use the "raw" type to access specific portions of files. If the server is unwilling to handle the requested type, it shall provide an error with status code 415 (unsupported media type).
drate	4	This field is typically used in conjunction with motion imagery that contains timing information such as MJ2 files. It may also be used for a timed sequence delivery of codestreams. This functionality is useful for LVSD systems where a sequence of still frames is collected at some number of frames/second. The "drate" request field specifies the clients desired delivery rate of frames. The rate is specified as a multiplicative factor relative to the original source rate. The "drate" request may be used to speed up or slow down delivery. Note a client may use a "drate" request in conjunction with an "srate" request (see "srate" above).
Cache Management Request Fields		
model	4	This field may be used in either stateless requests or in sessions to inform the server that it should add to or subtract from the server's model of the client's cache. For instance, when used with stateless requests, this provides a mechanism for the client to inform the server of its current cache contents, by adding all its cached data-bins to the initially empty model. For a session, the client may ask the server to subtract from its cache model if the client is running low on memory and needs to discard an entry from the cache. In general, the server must honour any request to subtract from the cache, but it may at its discretion ignore requests to add to the cache. The reasoning behind this is that adding to the cache simply prevents the transmission of redundant data, but this is only an efficiency issue – there is no error in sending redundant data. However, servers are expected to reasonably honour additive cache model requests so as to minimize network bandwidth usage.
tpmodel	0	This field is only used with JPT-streams, and thus servers are not required to support it. PJPIP recommends that implementations support JPT-

JPIP Request Field	Server Support Required	Notes
		streams.
need	4	This field is used only in stateless requests, and provides a mechanism for the client to inform the server about the set of data-bins that are of potential interest to the client. By default, if not provided, all data-bins are of potential interest (constrained by the region of interest if specified). The server should use this information to avoid sending information that is not of potential interest. One potential use of this field would be for a client that explicitly manages data-bins to request exactly those data-bins that it needs for a specific task (in conjunction with a full-resolution frame and region size so as not to impose any region constraints).
tpneed	0	This field is only used with JPT-streams, and thus servers are not required to support it. PJPIP recommends that implementations support JPT-streams.
mset	4	This field provides a mechanism to manage the cache model over a series of codestreams. It may be useful, for instance, in motion imagery applications and LVSD applications so that the client can easily inform the server when it discards an earlier codestream from its cache, in case the client later "rewinds" to that previous codestream and would expect the server to retransmit data.
Upload Request Fields		
upload	0	This field may be used to upload an image to the server. It is envisioned that other mechanisms will be used to upload images to a PJPIP server, so this feature is not required. In fact, in many cases, it may be required that servers do not allow this capability (or at least implement appropriate access privileges to restrict its availability).
Client Capability and Preference Request Fields		
cap	1	While a client may optionally specify its capabilities using this field, the server is not required to take any specific action based on these capabilities. PJPIP servers may simply ignore this field (without issuing any error).
pref	2	PJPIP servers are not required to support any client preferences, but some of these may be useful for various scenarios. For instance, selection of sequential vs. reverse-sequential codestream sequencing may be useful for LVSD collections to provide forward/reverse functionality. In addition, the max bandwidth and/or bandwidth slice preferences may be useful in managing the available bandwidth in constrained applications. The JPIP PDAM4 Version 2 has added a strictness field to this request that allows clients to tell servers to minimize the amount of additional data sent. If a requested preference is not supported, it shall simply be ignored, unless the client indicates that the preference is required using the "/r" flag, in which case the server must issue an unsupported preference response along with an error status code 501 (not implemented).
csf	1	This field may be used by clients to specify the contrast sensitivity function of the display and viewer. An enhanced server may be able to use this information to deliver information in an order that is optimized for the viewing conditions, but this is not required. If the server does not support any special handling based on the contrast sensitivity function, it should merely ignore this field without issuing an error.
mctres	0	PJPIP servers are not required to support the Multiple Component Transform (MCT) framework out of ISO/IEC 15444-2. PJPIP servers should respond with code 501 (not implemented) if this request is received

JPIP Request Field	Server Support Required	Notes
		from a client. A future version of this Recommended Practice may require support for this function.

Table 4 – Summary of Client Request Fields

6.2.2 Server Reply Syntax

The server's response to a client's request consists of the following elements:

- Status code: A 3-digit integer code. If the server accepts the request and delivers data in response to it, it shall use the status code of 200. JPIP makes use of a subset of the HTTP/1.1 codes found in RFC 2616. See Annex D.1.3 of ISO/IEC 15444-9 for descriptions of all the status codes.
- Reason phrase: A textual description of the status. Although the exact syntax of the reason phrase is left to the server's discretion, typically the status code 200 will be described as "OK" or "OK, with modifications" depending on whether any of the requested fields were modified by the server. For any of the error codes, the server should provide a meaningful description of the problem as part of the reason phrase to assist in troubleshooting potential client problems.
- JPIP response header: Describes any changes to the requested parameters that the server has made and various other parameters that may be required. The names of the response headers are derived from the corresponding requests (with a "JPIP-" prefix). The server's reply may include several response headers.
- Response data: The actual data from the target that is included in servicing the response. For the JPP- or JPT-stream return types, this consists of a series of data-bin messages.

6.2.3 Server Response Headers

Table 5 describes the response headers used by the JPIP server in response to requests from a JPIP client. The "Notes" column is not intended as a complete restatement of the contents of Annex D.2 but rather as additional information for JPIP implementers, including any particular requirements imposed by PJPIP.

The key to the server support requirement level is as follows:

- 0: Not required if the corresponding optional client request field is not implemented (or not required at all)
- 1: May be required if the corresponding optional client request field is not implemented.
- 2: May be required if the corresponding partially required client request field is not fully supported.
- 3: Required if the server makes a change to the parameter value specified by the client in the corresponding request field.
- 4: Always required if the client request includes the corresponding request field (or other specific conditions as described in the notes).

JPIP Response Header	Server Support Required	Notes
Target Identification Response Headers		
JPIP-tid	4	Required if the target ID served differs from that requested or if the client specified a target ID of 0 (or did not specify a target ID request field). PJPIP servers are required to generate unique target IDs for each logical target and a new ID must be assigned if the logical target is modified in any way. Typically, this unique target ID will be a function of the file name, modification date/time, and a unique identification of the server (such as its hardware address). PJPIP servers shall not return a target ID value of 0.
Session and Channel Response Headers		
JPIP-cnew	4	Required if (and only if) the server assigns a new channel in request to the "cnew" request field. If the server is unwilling to create the requested new channel (e.g., if the server's maximum number of channels per session have been exhausted), it must not send this response header.
JPIP-qid	4	Required if the request included a "qid" field, in which case the returned value shall be identical to the "qid" value in the request.
View-Window Response Headers		
JPIP-fsiz	3	Required if the frame size served differs from that requested.
JPIP-rsiz	3	Required if the region size served differs from that requested.
JPIP-roff	3	Required if the region offset served differs from that requested
JPIP-comps	3	Required if the range of component indexes served differs from that requested.
JPIP-stream	3	Required if the codestream indexes served differ from those requested via either the codestream request field or the context request field.
JPIP-context	4	Required if the server is able to process any of the context range values, in which case it identifies the actual codestream indexes associated with the context range.
JPIP-roi	4	If the server supports a requested named ROI, it shall specify the actual frame size, region size, and offset that correspond to this region of interest. If the server does not support named ROIs or cannot fulfill the ROI request, it shall reply with "JPIP-roi: roi=no-roi".
JPIP-layers	3	Required if the number of layers served is less than that requested. This could happen, for instance, in response to an alignment request field.
JPIP-srate	4	If the average sampling rate is expected to differ from that requested, the server shall send this response field. PJPIP servers are required to support the sampling rate request field.
Metadata Response Headers		
JPIP-metareq	0	If the server supports the metadata request field, it should send this response if the metadata served differs from that requested. Metadata requests are not relevant to raw codestreams, and PJPIP servers are not required to support the metadata request field.
Data Limiting Response Headers		
JPIP-len	3	Required if the server had to send more than the requested byte limit in order to allow a non-empty response. In this case, the returned

		value indicates a suitable maximum response length for subsequent requests (which may be greater than the actual response length for the current request).
JPIP-quality	1	Although the server is not required to be able to interpret the requested quality value, it must at least return a value of "-1" if it ignored the client's request.
Server Control Response Headers		
JPIP-type	4	For the HTTP and HTTP-TCP transport, this response header is optional if the HTTP "Content-Type:" header indicates the image return type. Otherwise, it would be required as part of the first response associated with a session, but would be optional for later requests within the same session.
Cache Management Response Headers		
JPIP-mset	3	Required if the server's cache model set differs from that requested. At its discretion, the server may model less than requested (possibly leading to redundant data transmission), but it is not allowed to model more than requested (as this could cause required data to be omitted).
Client Capability and Preference Response Headers		
JPIP-cap	0	This response header may be used to indicate capabilities that the client needs in order to properly handle the returned image data. Servers are not required to provide these details.
JPIP-pref	2	This response header is used to describe any unavailable preferences. It is provided if a requested client preference is specified as required but is not supported by the server. In this case, the server should also return an error status code of 501 (not implemented). This response header should not be provided for any preferences that are requested without the "/r" (required) flag.

Table 5 – Summary of Server Response Headers

6.3 Server Region of Interest Modifications

PJPIP servers are *not required* to return all codestream data corresponding to a client's view-window request. The server has full control over modifying the requested region of interest and may deliver a smaller region than requested at its discretion. Under normal operating conditions, PJPIP servers are expected to honor all reasonable window requests. For example, a server which is typically used for delivery of images for interactive viewing on a desktop monitor may choose to limit the region of interest to 2k x 2k pixels with three color components. If a client truly needs more data, it may make subsequent requests for smaller windows until it receives all the data required. Under extreme conditions of network loading, it may be necessary for the server to limit requests to something smaller than this nominal threshold, and how this should happen is beyond the scope of this profile.

7 Using JPIP with Image Libraries (informative)

JPIP represents a fundamental change in the delivery of image data. Traditionally image libraries (and web sites) have followed an "FTP" paradigm where the entire image file is transferred from server to client and the client then displays or saves off the image. While this type of operation is possible with JPIP, it is not why the protocol was created. JPIP was created to interactively send portions of a JPEG 2000 codestream from server to client based on the client's requests (i.e. user

input). The question of where JPIP functionality should be applied within an imagery dissemination architecture naturally comes to mind. Should image libraries be given JPIP server capability? Should a dedicated JPIP server be fielded instead? How will JPIP impact information flow within the architecture?

Answering these questions is beyond the scope of this Recommended Practice. However, the Open Geospatial Consortium has conducted experiments in marrying a JPIP server with a Web Coverage Service (see *WCS Implementation Specification Ver. 1.1.0* and the *OWS-4 IPR for WCS Support for JPEG 2000*). WCS was used to provide a mechanism for clients to request imagery and JPIP streams. Figure 3 illustrates how WCS was used to disseminate JPIP streams. First the client queries a catalog service (Image Catalog) for a list of WCS images that match a specific latitude and longitude (1). Next, the WCS client sends a WCS server a request for a specific coverage and receives the URL address of the JPIP server serving the image (2). Finally the JPIP client sends a JPIP request to the JPIP server which returns the portion of the requested image to the analyst (3).

The model chosen for the WCS/JPIP experiments is a good model for incorporating JPIP into an imagery dissemination architecture. It helps minimize the amount of software coding that must be done on existing image libraries and separates the JPIP exchanges from other information exchanges. The amount of communication between a JPIP server and its clients can be significant so dedicating servers to supporting JPIP exchanges is a good choice. This paradigm could be followed for other image servers beside WCS servers (e.g. NSILI, STANAG 4559 servers or IPLs).

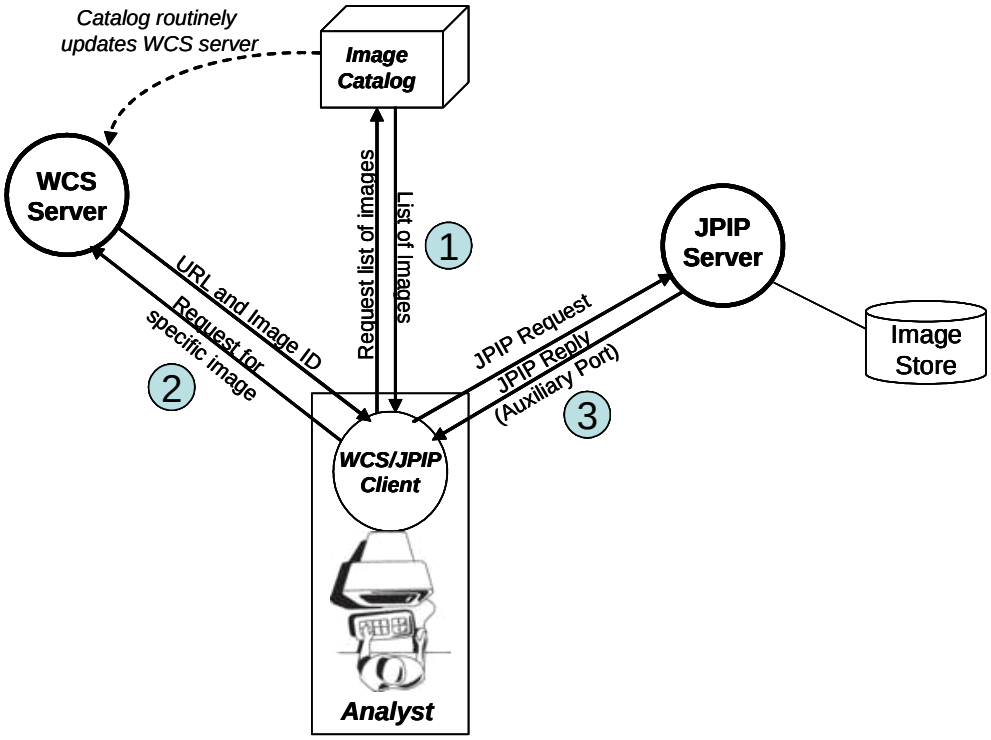

Figure 3 – Overview of how WCS will identify and deliver JPIP streaming data